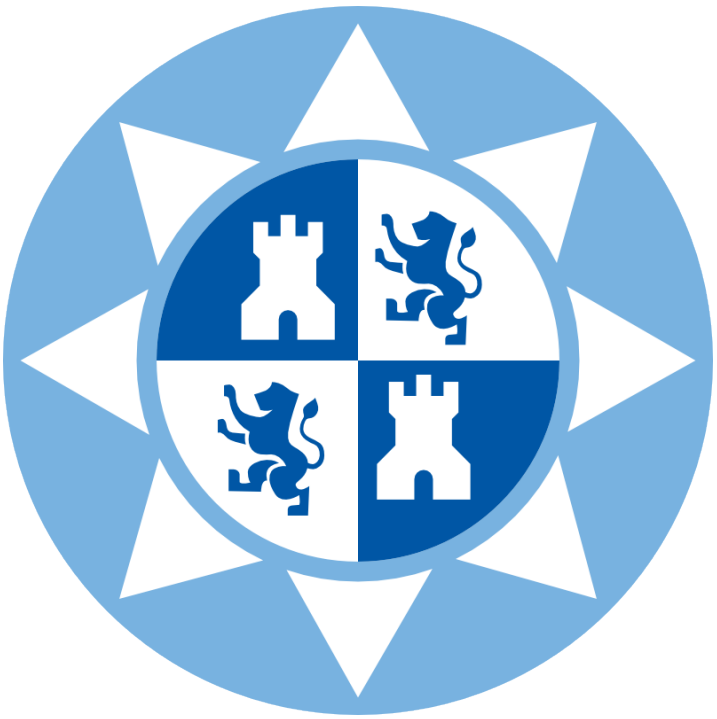
Universidad Politécnica de Cartagena

Escuela Técnica Superior de Ingeniería de Telecomunicación



Proyecto Fin de Carrera:

Aplicación para Android para el control de un vehículo a través de un trazo

Curso: 2012/2013
Titulación: Ingeniero de Telecomunicación
Autor: Juan Francisco Sánchez López
Director: Juan Carlos Sánchez Aarnoutse
Juan José Alcaraz Espín

Índice General de Capítulos:

Índice General de Figuras:

Índice General de Tablas:

Capítulo 1: Introducción y objetivos

1.1. Introducción

Los dispositivos móviles han supuesto una verdadera revolución en la vida de las personas, han cambiado nuestra forma de comunicarnos, nuestra forma de divertirnos, de jugar, de buscar, de aprender…

Su uso ha variado de manera significativa desde su aparición. Ya no son aquellos dispositivos que se utilizaban sólo para poder llamar o mandar mensajes, sino todo lo contrario, se han convertido en un ordenador personal que podemos llevar siempre con nosotros.

Con ellos podemos navegar por internet desde cualquier sitio, escuchar música, tomar fotos, reproducir videos, mandar emails, *whatsapps* y disfrutar de un sin fin de aplicaciones, que cada día añaden mas funcionalidad a los dispositivos, todo ello haciendo uso de diversas tecnologías como pueden ser UMTS, Wifi, Bluetooth, GPS, NFC...

El punto de inflexión se produce en el año 2007, con el lanzamiento del *iPhone* por parte de Apple, con su sistema operativo *iOS*. No es que antes del iPhone no hubiesen teléfonos inteligentes, que los había, pero si que cambian muchas cosas.

Podemos afirmar que nacen los smartphone, como los conocemos hoy en día. Se modifica la forma de interactuar con el dispositivo, se emplea una pantalla táctil que ocupa la totalidad del teléfono, en vez de botones, lo cual hace el uso mas intuitivo; nace el mercado de las aplicaciones, *apps*, semejantes a los programas que utiliza un ordenador.

Un año mas tarde, en el 2008, llegó la competencia, y con ella *Android*, un sistema operativo para dispositivos móviles cuya mayor parte está bajo licencia de software libre Apache. Con lo cual es ideal para instituciones educativas, empresas o simples usuarios puedan realizar experimentos sin tener que pagar por las licencias.

1.2. Objetivos

El proyecto que se va a tratar en este documento surge a partir de la idea de poder controlar un coche de carreras radiocontrol desde un dispositivo móvil, haciendo uso para ello de la tecnología inalámbrica Bluetooth.

Figura 1: Coche Radiocontrol

Debido a la multitud de ejemplos similares que se pueden encontrar en el mercado, decidimos introducir una dificultad añadida para innovar en el sector. En lugar de controlar el movimiento del coche radiocontrol en una aplicación con un sistema simple de botones (delante, atrás, izquierda y derecha) cambiamos la interfaz por una pantalla donde el usuario tiene que dibujar un determinado movimiento, que después el coche se encargará de reproducir.

Basándonos en estas características hemos desarrollado una aplicación para la plataforma Android, cuyo objetivo consistirá en dirigir nuestro coche de carreras mediante un teléfono inteligente o una tableta, simplemente dibujando una figura en nuestra pantalla, sustituyendo así por completo el mando radiocontrol de nuestro dispositivo.

Para la conexión entre el dispositivo móvil y el coche de carreras, utilizamos un módulo Bluetooth, el cual se encarga de enviar los datos de forma inalámbrica.

Mientras que en el coche radiocontrol, instalamos un microcontrolador, Arduino, que será el que transforme las líneas de código que le llegan desde el dispositivo móvil en señales eléctricas que moverán el motor y las ruedas de nuestro vehículo.

Por tanto, los objetivos que persigue nuestro proyecto son los siguientes:

- Familiarizarse con el entorno de desarrollo Arduino.
- Controlar una conexión Bluetooth entre el microcontrolador Arduino y un dispositivo móvil.
- Desarrollar la aplicación móvil, que nos permita pintar sobre la pantalla un determinado recorrido.

Además de objetivos secundarios, como pueden ser:

- Aprender a programar aplicaciones para el microcontrolador Arduino.
- Aprender a programar aplicaciones para el sistema operativo Android, y conocer los detalles de su arquitectura, lo cual nos permitirá desarrollar nuestra aplicación.
- Dar los pasos necesarios para poder llegar a publicar la aplicación en el Google PlayStore.

1.3. Estructura de la memoria

En este apartado explicamos de forma breve el contenido de cada uno de los capítulos que conforman esta memoria.

En el *Capítulo 1, Introducción y Objetivos*, se expone de forma clara un breve introducción del proyecto, los objetivos del mismo y una visión general de lo expuesto en esta memoria.

En el *Capítulo 2, Tecnologías empleadas*, detallamos cada uno de los elementos utilizado en este proyecto, y relación con los demás.

En el *Capítulo 3, Implementación de la plataforma con Arduino,* nos encargaremos de explicar la parte Hardware del proyecto.

En el *Capítulo 4, Implementación de la App*, aquí nos centraremos en la parte Software del proyecto, como hemos desarrollado nuestra aplicación Android, los pasos necesarios

En el *Capítulo 5, Desarrollo desde el punto de vista de usuario*, se detalla como manejar la aplicación desde cualquier dispositivo Android.

En el *Capítulo 6, Conclusiones y líneas futuras*, se reflexiona sobre los objetivos conseguidos, las dificultades encontradas y se indican posibles líneas de investigación futuras.

Por último tenemos la *Bibliografía*, donde se exponen las referencias consultadas para la elaboración y redacción del proyecto.

Capítulo 2: Tecnologías empleadas

2.1. Introducción

En este capítulo vamos a ver las tecnologías usadas para la realización del proyecto, intentando explicar cada una de ellas de forma simple y clara, para poder tener una visión más clara de la elaboración del proyecto.

2.2. Arduino

¿Qué es Arduino?

Arduino es una plataforma de desarrollo de código abierto, libre de licencias, basada en un placa con un sencillo microcontrolador y un entorno de desarrollo que nos permite crear nuestros propios programas, para ejecutarlos luego en la placa.

Los microcontroladores siguen la arquitectura de AVR, que son una familia de microcontroladores RISC de Atmel. El microcontrolador se programa mediante el lenguaje de programación Arduino, que está basado en *Wiring*, un *framework* de programación libre para microcontroladores.

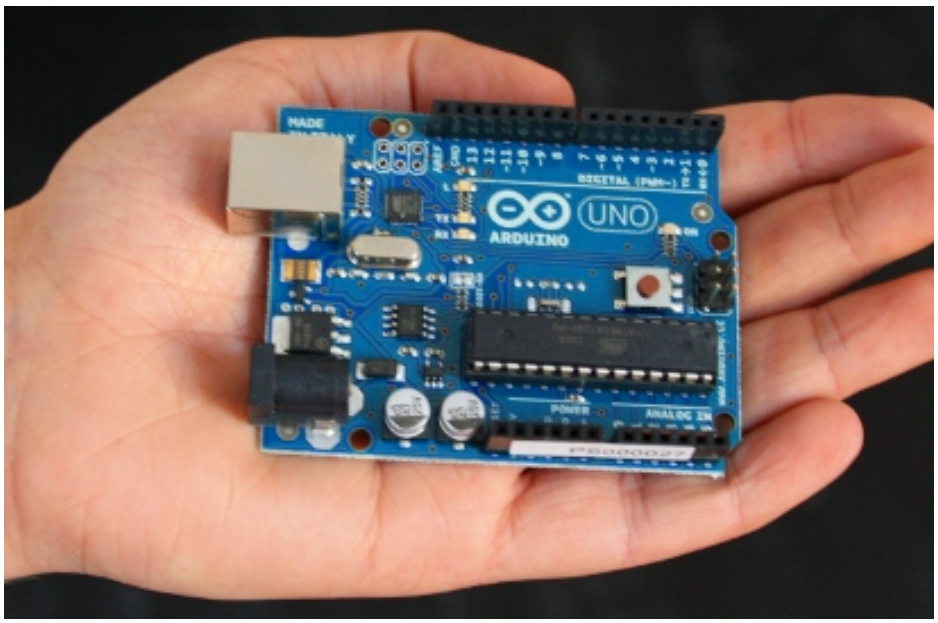


Figura 2: Arduino

El entorno de desarrollo integrado, en inglés IDE (Integrated Development Environment), es un software libre, que se puede instalar en cualquier ordenador (Windows, Mac OS X o GNU/Linux) y que podemos descargar desde la propia pagina de Arduino.

Implementa el lenguaje de programación *Processing*, que es un lenguaje de programación de código abierto que está basado en Java, lo que le permite heredar todas sus funcionalidades, convirtiéndose en una poderosa herramienta que nos permite realizar proyectos complejos.

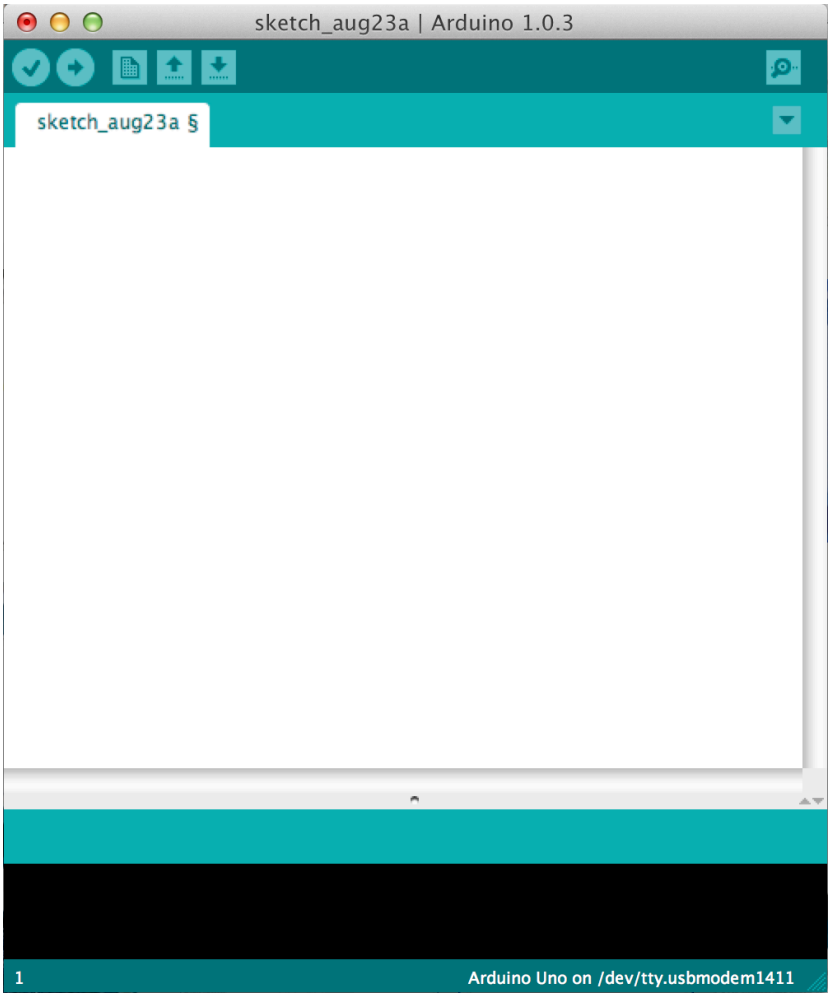


Figura 3: IDE Arduino

La placa que vamos a utilizar en nuestro proyecto es la Arduino UNO, la cual está basada en el microcontrolador ATMega328. Posee 14 pines de E/S, 6 entradas analógicas, un resonador cerámico (16MHz), una conexión USB, una entrada de alimentación, cabeceras ICSP y un botón de reset.

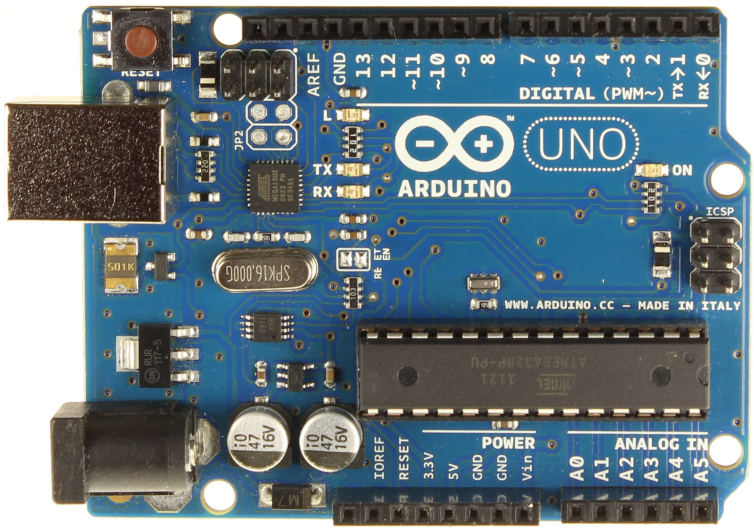


Figura 4: Placa Arduino UNO

Para hacerla funcionar simplemente tenemos que conectarla a un ordenador mediante la entrada USB, o alimentarla mediante el uso de una batería o un adaptador de corriente continua.

Arduino Uno se diferencia de las placas predecesoras en que no hace uso del driver FTDI USB-to-serial, sino que en su lugar cuenta con el ATMega16U2 programado como un convertidor USB-to-serial.

Las principales características de la placa son:

Microcontrolador	ATMega328
Voltaje Operativo	5 V
Voltaje entrada(recomendado)	7-12 V
Voltaje entrada (límite)	6-20 V
Pines E/S digitales	14 (6 pueden generar salidas PWM)
Pines de entrada analógica	6
Corriente pines E/S	40 mA
Corriente pin 3.3V	50 mA
Memoria Flash	32 KB (0.5 KB utilizados bootloader)
SRAM	2 KB
EEPROM	1 KB
Velocidad de reloj	16 MHz

Tabla 1: Características Arduino UNO

2.3. Bluetooth

¿Qué es Bluetooth?

Bluetooth es una tecnología inalámbrica cuyo objetivo es simplificar la comunicación entre dispositivos informáticos, como por ejemplo la conexión entre un ordenador y dispositivo móvil.

Esta tecnología fue desarrollada en 1994 por Jaap Haartsen y Mattisson en los laboratorios Ericsson, Suecia. Estaban trabajando en un proyecto que les permitiese poder cambiar las diapositivas de una presentación mediante un teléfono móvil.

Mas tarde Ericsson se unió a otras compañías para formar el Bluetooth Special Interest Group (SIG), el cual tenia como objetivo desarrollar la tecnología para que se convirtiese en estándar abierto. Hoy en día el SIG cuenta con mas de 14.000 empresas de todo el mundo.

Especificaciones

Estamos hablando de una tecnología de ondas de radio de corto alcance, 2.4GHz, que fue pensada para dispositivos de bajo consumo como dispositivos móviles, que requieren un corto alcance de emisión y basados en transceptores de bajo costo.

Los dispositivos que incorporan esta tecnología pueden comunicarse entre ellos, y no hace falta que estén alineados, ni en la misma habitación siempre que se encuentre dentro de su alcance. Los podemos clasificar en función de su potencia de transmisión, siendo totalmente compatibles entre los diferentes tipos:

Clase	Potencia Máxima Permitida (mW)	Potencia Máxima Permitida (dBm)	Alcance (Aprox.)
Clase1	100 mW	20 dBm	100 m
Clase2	2.5 mW	4 dBm	10 m
Clase3	1 mW	0 dBm	1 m

Tabla 2: Clases Bluetooth

En la mayoría de los casos, la cobertura efectiva de un dispositivo de clase 2 se extiende cuando se conecta a un transceptor de clase 1. Lo que es debido a la mayor sensibilidad y potencia de transmisión del dispositivo de clase 1, es decir, la mayor potencia de transmisión del dispositivo de clase 1 permite que la señal llegue con energía suficiente hasta el de clase 2. Por otra parte al tener el dispositivo de clase 1 mayor sensibilidad permite recibir la señal del otro aunque esta sea más débil.

También los podemos clasificar en función de su ancho de banda:

Versión	Ancho de banda
Versión 1.2	1 Mbit/s
Versión 2.0 + EDR	3 Mbit/s
Versión 3.0 + HS	24 Mbit/s
Versión 4.0	24 Mbit/s

Tabla 3: Versiones Bluetooth

Curiosidad

El nombre Bluetooth procede del rey danés y noruego Harald Blåtand, que al traducirlo al inglés es Harald Bluetooth, fue conocido por unificar las tribus noruegas, suecas y danesas y por convertirlos al cristianismo.

También se cree que recibió tal apodo por haber padecido una rara enfermedad, la eritroblastosis fetal, que habría echo que alguno de sus dientes tuviera un color azulado.

El logo de Bluetooth es la superposición de las runas nórdicas que conforman las iniciales del nombre y apellidos de dicho rey. La ᚼ (Hagall) y la ᛒ (Berkana).

Figura 5: Logotipo Bluetooth

Modulo Bluetooth

El módulo bluetooth empleado en el proyecto es el *HC-06*, el cual utiliza el protocolo *UART RS232 Serial*, y es ideal para el desarrollo de aplicaciones inalámbricas, debido a su bajo coste:

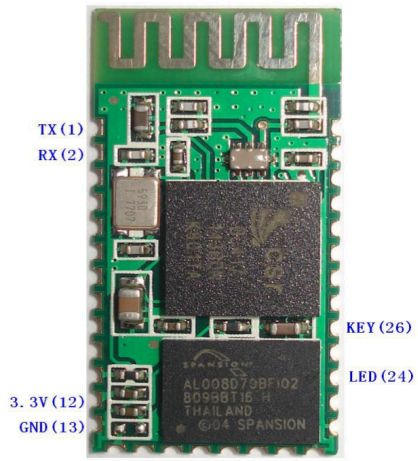


Figura 6: Módulo HC-06

Se puede encontrar el chip para soldarlo uno mismo, o ya montado sobre tarjetas de desarrollo (breakout), con los pines necesarios para realizar la comunicación:

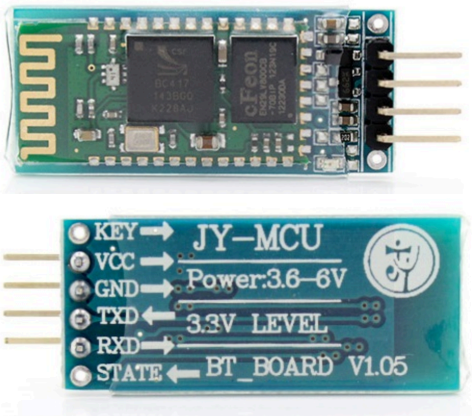


Figura 7: Módulo HC-06 montado

Como se aprecia en la figura de arriba, nuestro modulo únicamente dispone de 4pines:

- V_{CC} -> pin de alimentación
- GND -> pin de masa o tierra
- TXD -> pin de transmisión
- RXD -> pin de recepción

Las principales características del HC-06 son:

- Compatible con el protocolo Bluetooth V2.0
- Voltaje de alimentación: 3.3 V_{DC} – 6V_{DC}
- Voltaje de operación: 3.3 V_{DC}
- Baud Rate ajustable: 1200, 2400, 4800, 9600, 19200, 38400, 57600, 115200 baudios
- Tamaño: 4.4 cm x 1.6 cm x 0.7 cm
- Peso: 7 gramos
- Corriente de operación: < 40 mA
- Corriente modo sleep: < 1mA

Cabe destacar que el HC-06 tiene un hermano, el módulo HC-05, y que aparentemente son iguales (solo varía la conexión del pin KEY 26 HC-06 y 34 HC-05)

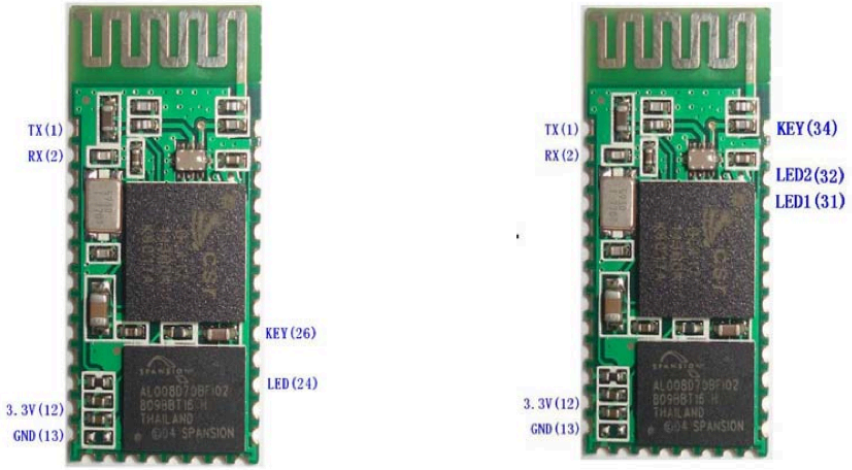


Figura 8: Comparación de módulos, HC-06 izquierda y HC-05 derecha

Pero la diferencia radica en que el módulo HC-06 solo funciona como *esclavo*, mientras que el HC-05 puede funcionar como *maestro/esclavo*.

Para reconocerlos basta con alimentarlo, y buscar el dispositivo bluetooth con un PC o un dispositivo móvil. El HC-06 será encontrado con el nombre de "*linvor*", mientras que el HC-05 aparece con el nombre de "HC-05".

Figura 9: Asistente BT del ordenador

2.4. Android

¿Qué es Android?

Android es un sistema operativo de código abierto orientado inicialmente a dispositivos móviles táctiles como *smartphones* o tablets, aunque en la actualidad podemos encontrarlo en una multitud de dispositivos: ordenadores, consolas, cámaras de fotos, relojes de pulsera, gafas de realidad aumentada, etc.

Fue desarrollado inicialmente por la empresa *Android Inc*. a la que Google respaldo financieramente hasta que la adquirió en 2005 decidió comprarla. Mas tarde en 2007, Android pasaría a ser el principal producto de la *Open Handset Alliance*, un consorcio de compañías desarrolladores de software y hardware, entre las que destacan: Intel, Samsung, LG, Motorola, T-Mobile, Qualcomm, etc. Y cuyo objetivo era desarrollar estándares abiertos para dispositivos móviles.

Google liberó la mayoría del código de Android bajo licencia Apache, una licencia libre y código abierto.

Figura 10: Logotipo Android

Especificaciones

Android está basado en una versión modificada del kernel de Linux 2.6, que fue lanzado al mercado en Octubre del 2008. Al estar basado el núcleo en Linux, el sistema es libre y gratuito. Cualquier desarrollador puede crear sus propias apps y probarlas en un terminal sin necesidad de pagar nada.

Android utiliza una variación del lenguaje de programación Java. En este lenguaje en el se trabaja con una maquina virtual, que se encarga de interpretar el código que genera nuestro programa y ejecutarlo. La ventaja es que solo tenemos que compilar las programas una vez, y la maquina virtual es capaz de interpretar y ejecutar el código compilado, el bytecode.

En Android tenemos *Dalvik*, que es una variación de la maquina virtual de Java, que se encarga de hacer un proceso similar, pero no es compatible con el bitcode Java. Es decir, cuando programamos las aplicaciones lo hacemos en Java, pero a la hora generar el ejecutable mediante el SDK de Android, este tendrá un extensión .dex, correspondiente a Dalvik, con lo que no podemos ejecutar aplicaciones Java en Android ni viceversa.

Arquitectura

La arquitectura del sistema operativo la podemos dividir en los siguientes niveles:

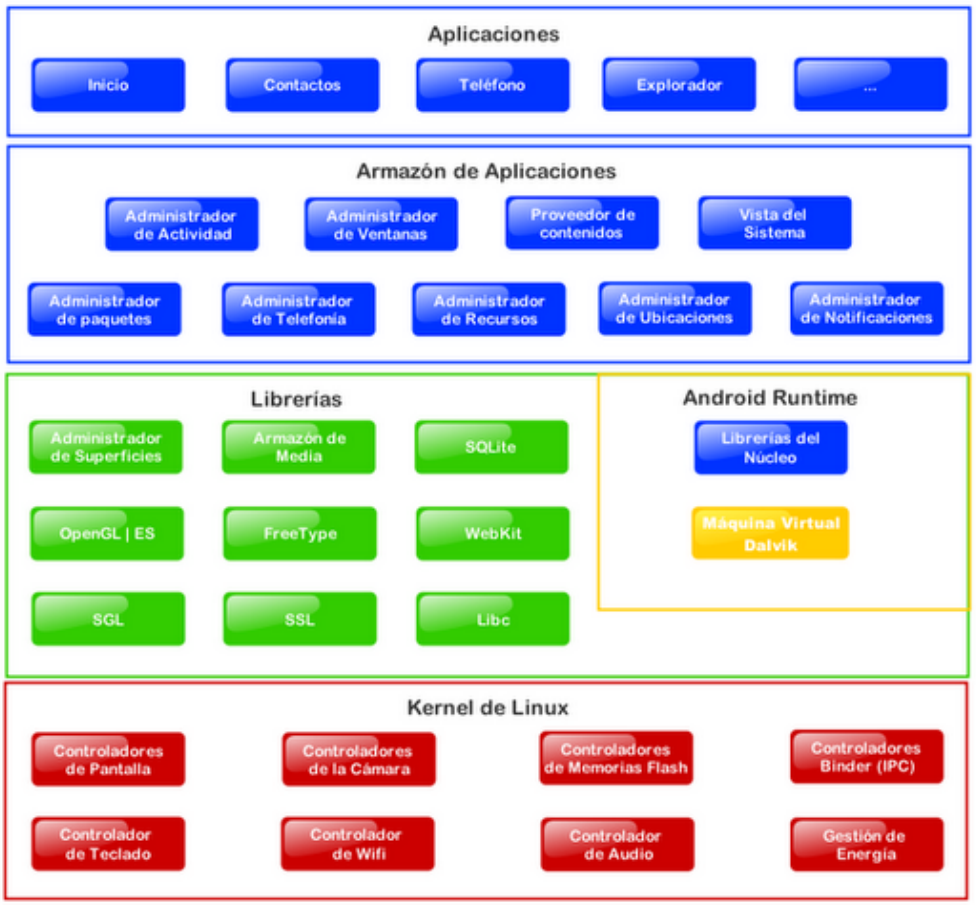


Figura 11: Arquitectura de Android

En el **nivel de aplicaciones** se encuentran todas las aplicaciones del dispositivo, las nativas (programadas en C/C++) y las que administradas (programadas en Java), las que vienen por defecto en nuestro teléfono móvil (cámara de fotos, galería, navegador, etc.) y también las que el usuario se descarga del PlayStore, de terceras empresas o que crea él personalmente.

En esta capa encontramos también la aplicación principal del sistema: Home (launcher), porque es la que nos permite ejecutar otras aplicaciones mediante una lista, o mostrando diferentes escritorios donde el usuario puede situar los accesos directos a aplicaciones que prefiera o incluso *widget*, que son aplicaciones que nos muestra información visualmente.

En el **nivel de framework de aplicaciones** representa el conjunto de herramientas, clases y servicios, que utilizan directamente las aplicaciones para realizar sus tareas. Todas las aplicaciones utilizan el mismo conjunto de API y el mismo framework representado en el este nivel. La mayoría de los componentes de este nivel son librerías Java que acceden a recurso de capas inferiores para poder realizar su trabajo

El siguiente nivel se corresponde con las **Librerías**, que son las bibliotecas nativas de Android, las cuales han sido desarrolladas en C/C++ y compiladas para el hardware específico de cada teléfono, por lo que normalmente el fabricante del dispositivo es el que se encarga de su desarrollo.

Las librerías ayudan a las aplicaciones a realizar las tareas habituales, evitando tener que codificarlas cada vez, haciéndolo mas eficiente. Algunas de las mas importantes son las siguientes: OpenGL (gráfica), Webkit (navegador), SQLite (Bases de datos) entre muchas otras.

Al mismo nivel encontramos el **Runtime Android**, o lo que es lo mismo, el entorno de ejecución de Android. No se considera capa estrictamente hablando, ya que esta formado por librerías, con una multitud de clases habituales de Java y otras especificas de Android. También está formado por la maquina virtual *Dalvik* que mencionamos anteriormente.

Y por último nos queda el **núcleo** del sistema, formado como ya hemos comentado por un kernel de Linux versión 2.6, similar al que se puede encontrar en una distribución normal de Linux, como Ubuntu, pero adaptado a las características del hardware que se encuentra en los dispositivos móviles.

El núcleo actúa como intermediario entre el hardware y el resto de los niveles. Para cada elemento de hardware hay un controlador que nos permite utilizarlo desde el software. Estos es así por que los desarrolladores no acceden a esta capa directamente, sino que se utilizan las librerías de los niveles superiores, con lo que nos ahorramos la necesidad de tener que conocer las características de cada teléfono. Si queremos usar la cámara, no accedemos a su hardware, sino que el sistema operativo se encarga de utilizar la que incluya el teléfono, sea cual sea.

Además el kernel también se encarga de la gestión de los diferente recursos de los que dispone el dispositivo y el sistema operativo, como pueden ser: la batería, la memoria RAM, el almacenamiento físico, networking, etc.

El verdadero potencial de Android reside en la personalización que nos ofrece, ya que el usuario tiene control total y puede del dispositivo un teléfono a su medida.

Curiosidad

El nombre del sistema operativo, Android hacen alusión a una novela de Philip Kindred Dick, *Do Androids Dream of Electric Sheep?,* escrito en 1968, y que posteriormente fue adaptada al cine con el nombre de *Blade Runner* (1982).

El nombre de los terminales de Google, los Nexus, también están relacionados con los androides que salen en la novela.

También es curioso destacar que todas las versiones de Android reciben su nombre de nombres de postres en ingles, siguiendo un orden alfabético:

Versión	Nombre
v1.0	**A**pple Pie
v1.1	**B**anana Bread
v1.5	**C**up Cake
v1.6	**D**onut
v2.0/2.1	**E**clair
v2.2	**F**royo
v2.3	**G**ingerbread
v3.0/3.1/3.2	**H**oneycomb
v4.0	**I**ce Cream Sandwich
v4.1/4.2/4.3	**J**elly Bean
v5.0	**K**ey Lime Pie

Tabla 4: Versiones Android

Capítulo 3: Implementación de la plataforma con Arduino

3.1. Introducción

En este capítulo nos centraremos en la parte hardware del proyecto, en como hemos modificado el coche radiocontrol, añadiéndole la placa de Arduino y el módulo bluetooth para poder dirigirlo desde nuestro dispositivo móvil.

3.2. Arduino

El código cargado en la placa Arduino es el siguiente:

```
/*
  PFC: BlueCar
  Autor: Juanfry
*/

// Vamos declarando los pines de salida de Arduino
int arriba      = 13;  // Pin 13 - Adelante
int izquierda  = 12;  // Pin 12 - Izquierda
int derecha    = 11;  // Pin 11  - Derecha
int abajo       = 10;  // Pin 10 - Atrás

void setup() {

Serial.begin(9600);    // Establecemos la velocidad 9600kbps

  // Inicializamos los pines como salidas
  pinMode (arriba, OUTPUT);
  pinMode (izquierda, OUTPUT);
  pinMode (derecha, OUTPUT);
  pinMode (abajo, OUTPUT);

}
```

```
void loop() {

    //Comprobamos si podemos recibir
    if ( Serial.available() > 0) {

     char cmd = Serial.read(); // Leemos el carácter recibido
     Serial.print(cmd);

      if(cmd == '1') {                  // Si recibimos un 1, nos movemos hacia delante

       digitalWrite(arriba, HIGH);
       digitalWrite (abajo, LOW);

      } else if (cmd == '2') {          // Si recibimos un 2, giramos a la izquierda

       digitalWrite (izquierda, HIGH);
       digitalWrite (derecha, LOW);

      } else if (cmd == '3') {          // Si recibimos un 3, giramos a la derecha

        digitalWrite (izquierda, LOW);
       digitalWrite (derecha, HIGH);

      } else if (cmd == '4') {          // Si recibimos un 4, nos movemos hacia atrás

       digitalWrite (arriba, LOW);
       digitalWrite (abajo, HIGH);

      } else if (cmd == '0') {      // Si recibimos un 0, ponemos todas las salidas a baja

       digitalWrite (arriba, LOW);
       digitalWrite (izquierda, LOW);
       digitalWrite (derecha, LOW);
       digitalWrite (abajo, LOW);
      }


 }


}
```

3.3. Conexión con el coche RC

El chip que emplea el coche radiocontrol es el RX2C, este chip es el que controla todo los movimientos del vehículo teledirigido, el cual podemos apreciar en la siguiente imagen tomada del interior del coche:

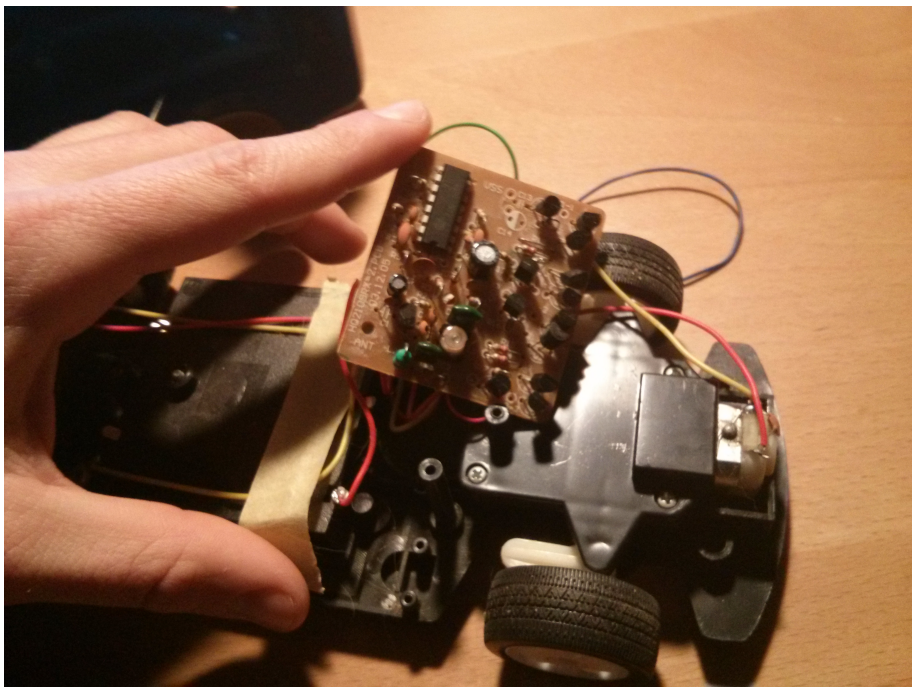


Figura 12: Conexión a la placa del coche RC

Lo que realizamos es puentear el chip RX2C con nuestra placa de Arduino de tal forma que ahora pase a ser esta última el cerebro de nuestro vehículo radiocontrol. Lo primero que necesitamos es mirar el datasheet del chip.

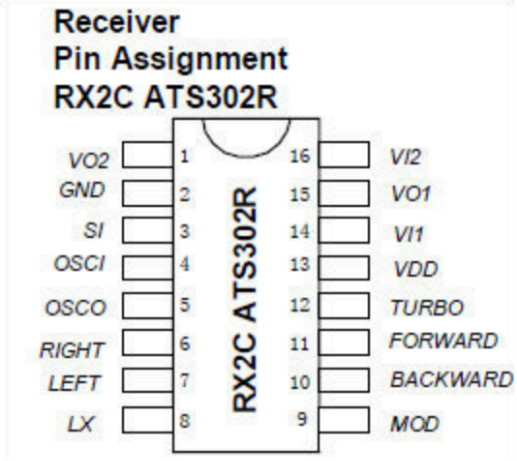


Figura 13: Chip Rx2C

Con lo que vemos que los pines que necesitamos puentear para que nuestro vehículo funcione son:

PIN	Movimiento
6	Derecha
7	Izquierda
10	Atrás
11	Delante

Tabla 5: Conexión de Pines

Por lo tanto las conexiones realizadas quedan de la siguiente manera:

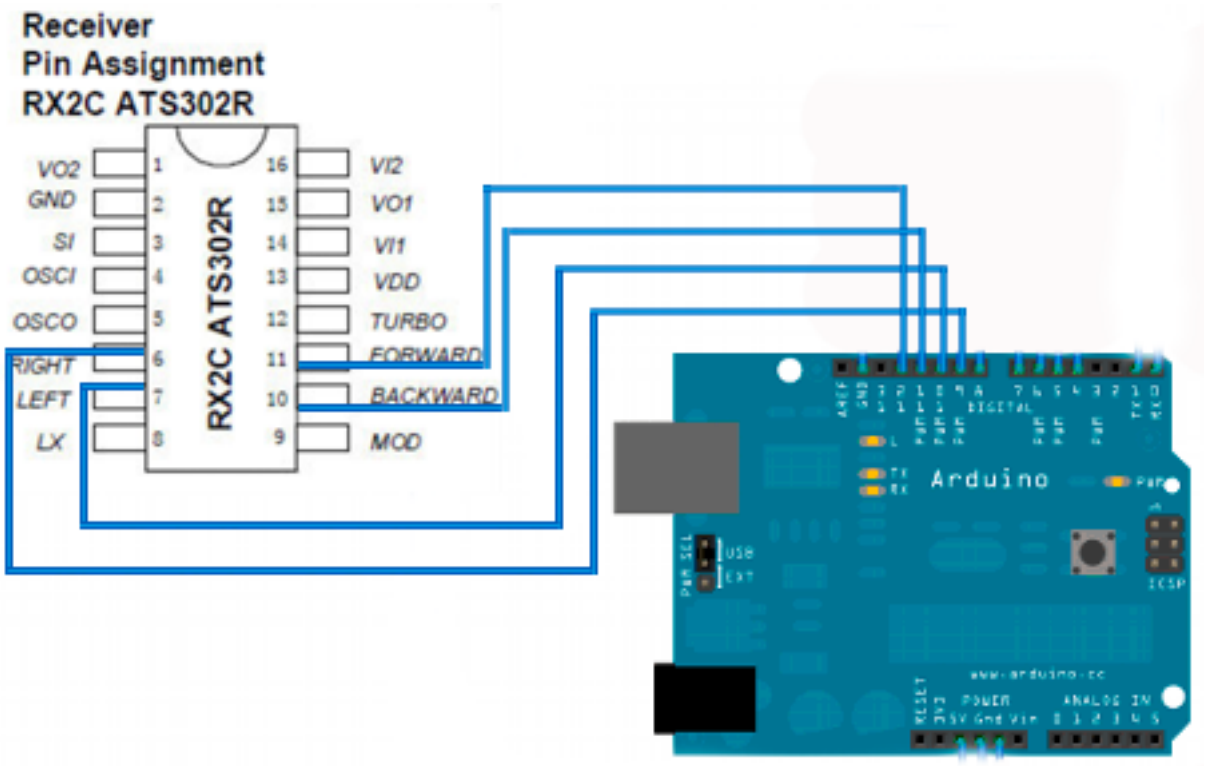


Figura 14: Conexión entre Arduino y RX2C

Donde hemos escogido los pines 9, 10, 11 y 12 como salida de la placa Arduino, ya que estos son los que configuramos en el código que cargamos en la placa, justo del apartado anterior.

3.4. Conexión con el módulo Bluetooth

Antes de realizar la conexión del módulo bluetooth, debemos configurarlo adecuadamente, y para ello hacemos uso de los comandos AT, con ellos podremos configurar algunas funciones como por ejemplo, cambiar el nombre del módulo, el código PIN para realizar la conexión o el Baud Rate.

Los comando son los siguientes:

- **AT**: sirve para comprobar la conexión, responde con OK.
- **AT+VERSION**: devuelve la versión del firmware del dispositivo.
- **AT+NAMEnombre:** para cambiar el nombre del dispositivo, responde con un **OKsetname**, está limitado a 20 caracteres.
- **AT+PINXXXX**: para cambiar el PIN de la conexión, responde con **OKserPIN**, por defecto será 1234.
- **AT+BAUDX**: para modificar el baud rate del dispositivo, X puede estar entre los siguiente valores:

Valor de X	Baud Rate
1	1200
2	2400
3	4800
4	9600 (Defecto)
5	19200
6	38400
7	57600
8	115200
9	230400
A	460800
B	921600
C	1382400

Tabla 6: Baud Rate

Para realizar la configuración del módulo cargamos el siguiente programa en nuestro controlador Arduino:

```
void setup(){

        Serial.begin(9600);
        delay(5000);

        Serial3.print("AT"); //Espera 1s según datasheet entre envio de comandos AT
        delay(1000);

        //Cambio de nombre
        Serial3.print("AT+NAMEBlueCar");
        delay(1000);

        //Configuracion del Baud Rate
        Serial3.print("AT+BAUD4");
        delay(1000);

        //Configuracion Password
        Serial3.print("AT+PIN1234");
        delay(1000);

        //Mostramos la finalizacion de la configuracion AT
        Serial.print("TERMINADO");
}
void loop() {

        //Nothing

}
```

Como muestra la siguiente imagen:

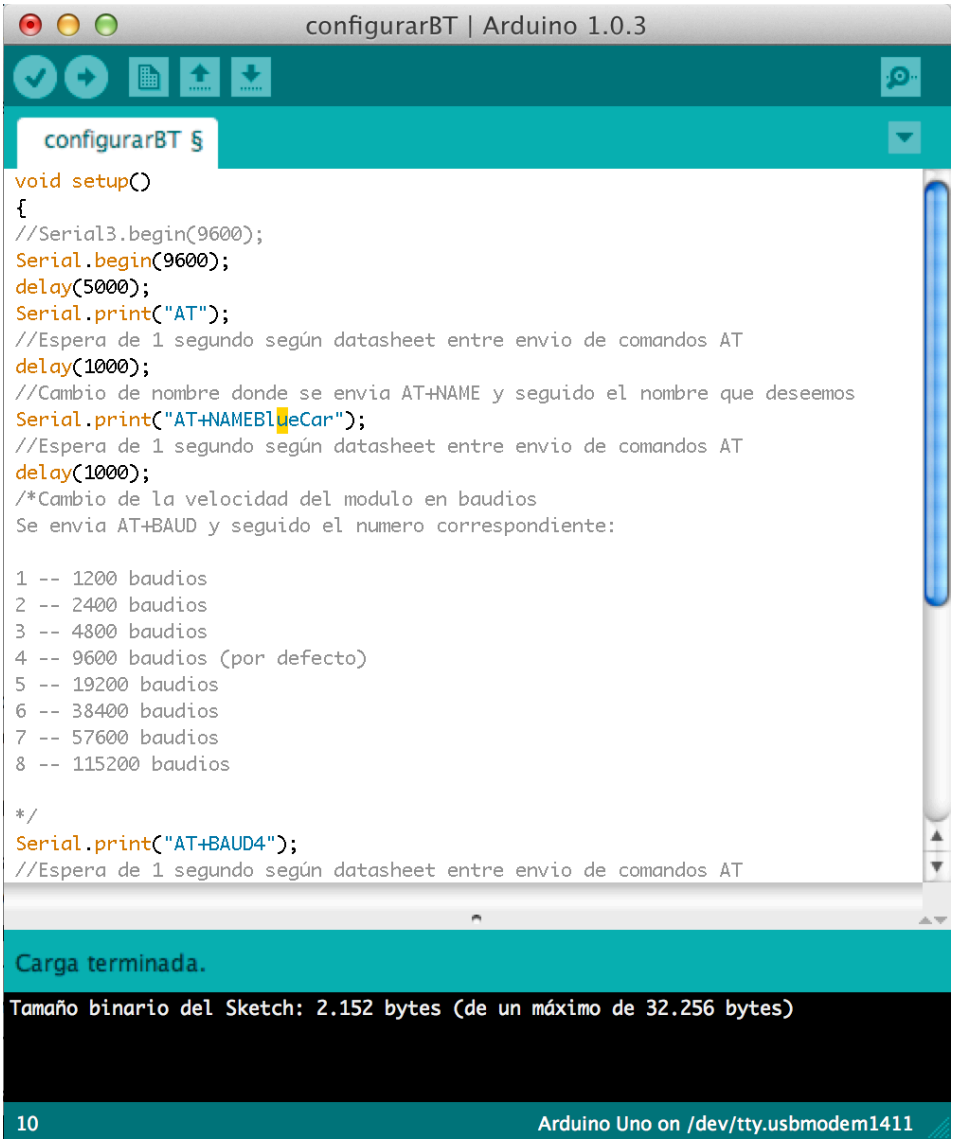


Figura 15: Configuración del módulo BT

Y conectamos la placa al módulo, solo son necesarias 4 conexiones:

- La alimentación lo conectamos a la salida de 5V del arduino
- La masa a la masa
- Intercambiamos el TX del módulo con el RX del arduino
- Y el RX del módulo con el TX del Arduino

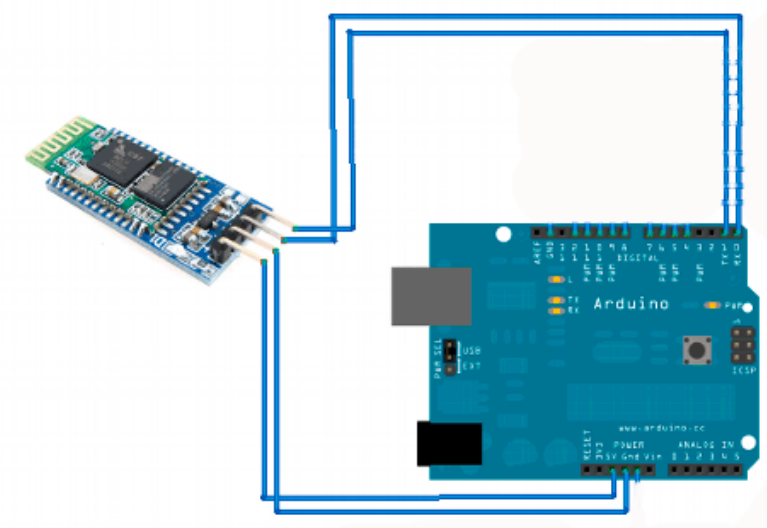


Figura 16: Conexión del módulo Bluetooth

El controlador está alimentado por el propio PC, y con ayuda del Monitor *Serial* del propio Arduino, comprobamos que los comandos AT se realizan correctamente.

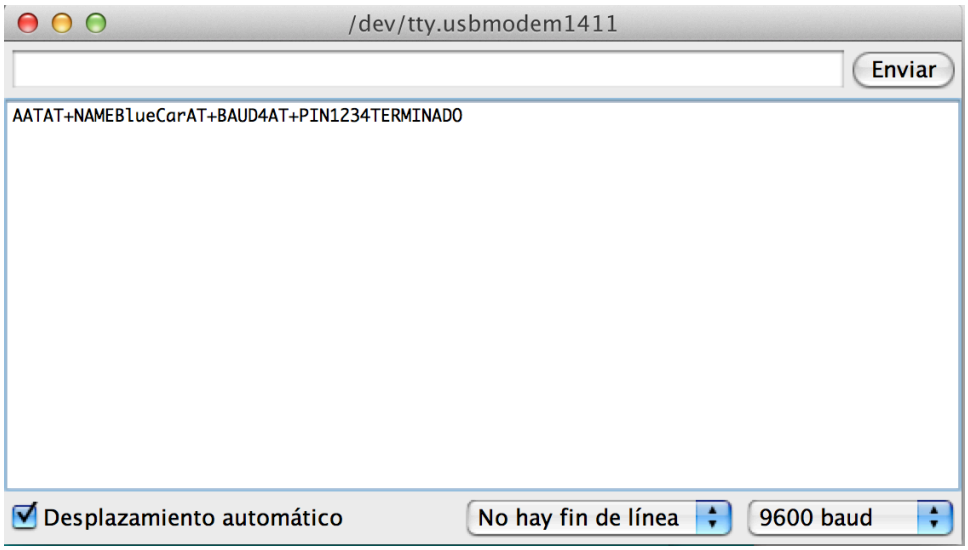


Figura 17: Terminal Serie

La siguiente imagen se corresponde con el montaje real realizado:

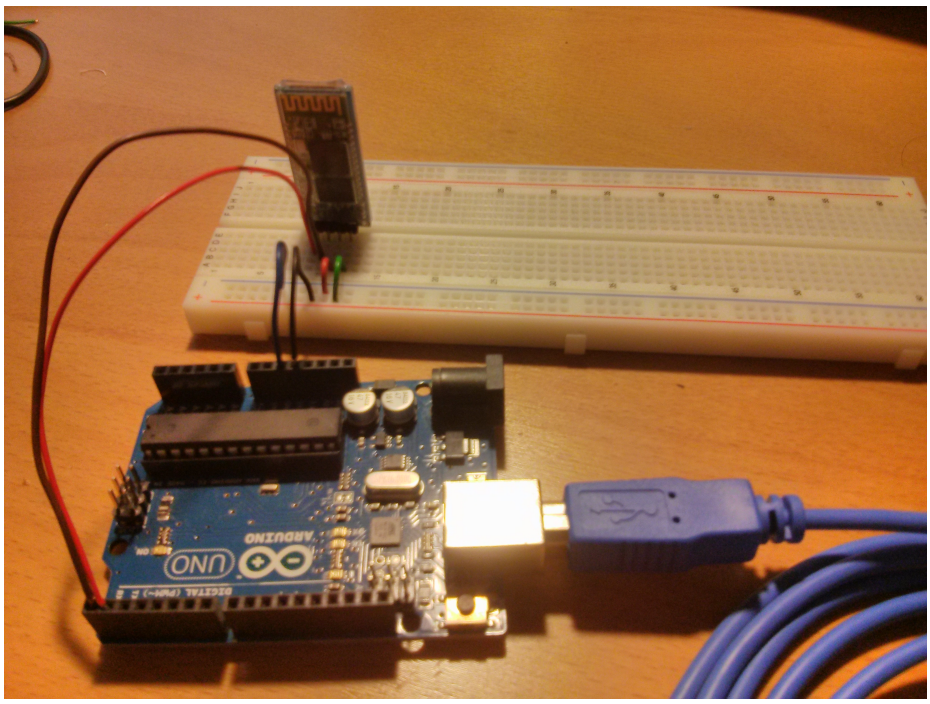


Figura 18: Montaje para la configuración bluetooth

Podemos comprobar que la configuración se ha realizado correctamente, porque ahora el ordenador ya no detecta un módulo Bluetooth denominado "*linvor*", sino que el módulo ahora aparece con el nombre que le hemos asignado.

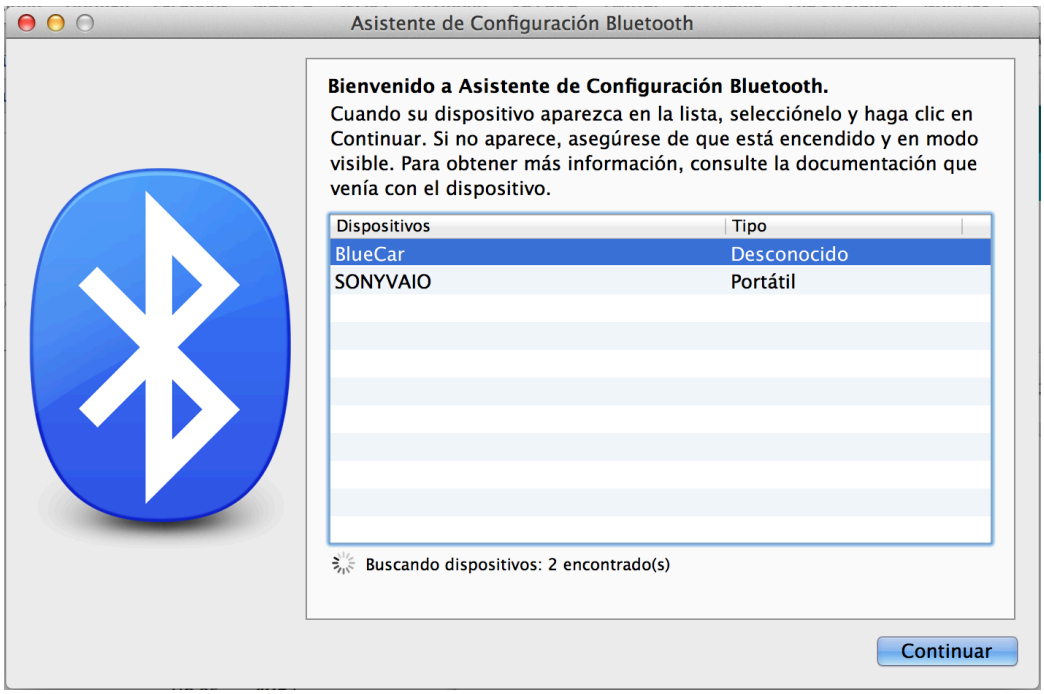
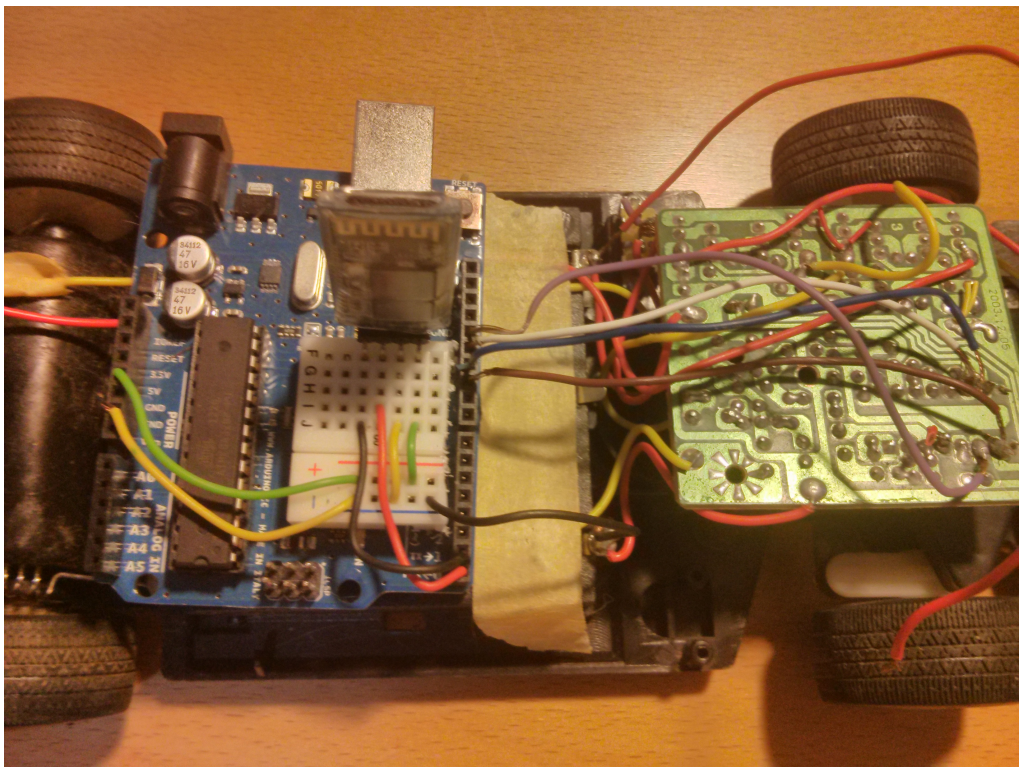


Figura 19: Bluetooth del ordenador

Si juntamos los dos montajes anteriores (Arduino-ChipRx2c y Arduino–MóduloBT) en uno solo, tendremos el montaje final:



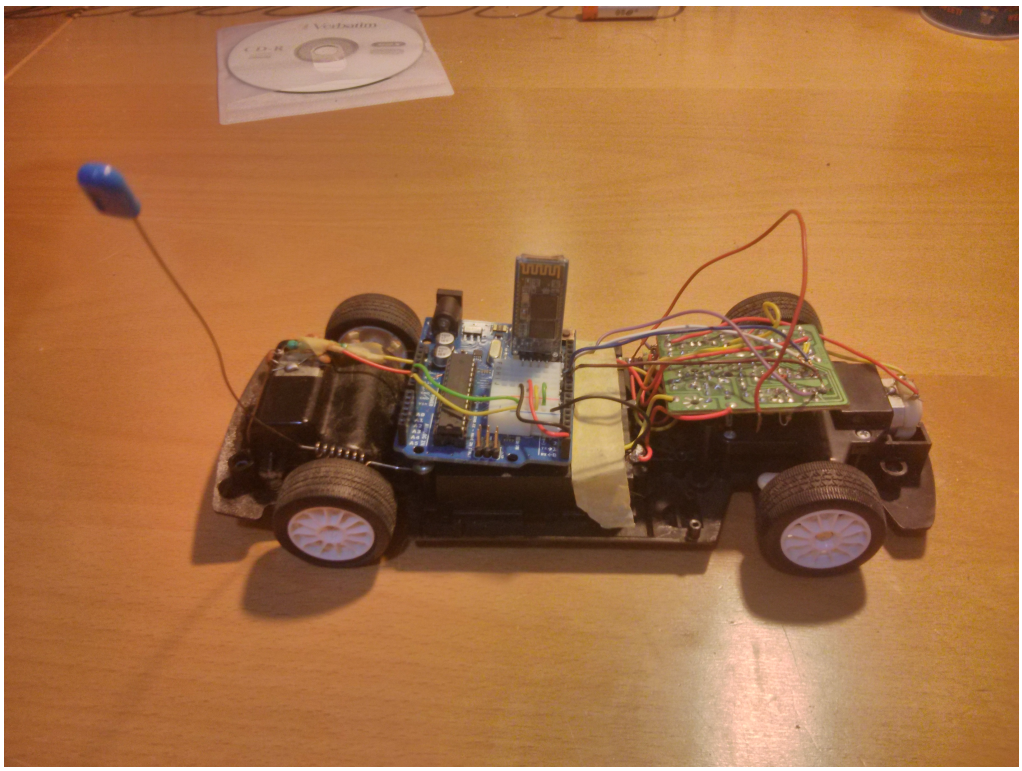


Figura 20 y 21: Montaje Final

Capítulo 4: Implementación de la aplicación

4.1. Introducción

Ahora nos adentraremos en la parte software del proyecto, en como hemos creado la aplicación de Android que realiza todo el proceso de comunicación con la parte hardware (Arduino) que ya vimos en el capítulo anterior.

4.2. Aplicación Android: Conexión Bluetooth

Lo primero que realizamos es la vinculación del dispositivo móvil con el vehículo, para ello habilitamos el bluetooth:

```
//Obtenemos el BlutoothAdapter, el cual es requerido en cualquier activityBluetooth
                mBluetoothAdapter = BluetoothAdapter.getDefaultAdapter();
                if (mBluetoothAdapter == null) {
                   //Dispositivo no soporta conex. Bluetooth, lo mostramos:
                        Toast.makeText(getBaseContext(), "Dispositivo no soporta conex.
Bluetooth", Toast.LENGTH_SHORT).show();
                }
                //Habilitamos la conex. Bluetooth en el caso de que no lo este
                if (!mBluetoothAdapter.isEnabled()) {
                   Intent enableBluetooth = new
Intent(BluetoothAdapter.ACTION_REQUEST_ENABLE);
                   startActivityForResult(enableBluetooth, 1);
                }
```

Iniciamos la Activity que nos permite encontrar los dispositivos bluetooth que hay a nuestro alrededor.

```
//Iniciamos DiscoveryActivity para buscar los dispositivos
        public void onBuscarClicked(View view){
                Intent intent = new Intent(this, DiscoveryActivity.class);
                startActivityForResult(intent, REQUEST_DISCOVERY);
        }
```

En esta nueva Activity escogemos el dispositivo con el que queremos vinvularnos, y regresamos a la MainActivity para realizar la conexión:

```
final BluetoothDevice device = data.getParcelableExtra(BluetoothDevice.EXTRA_DEVICE);

                    BluetoothSocket tmp = null;
                mmDevice = device;

                    // Get a BluetoothSocket to connect with the given BluetoothDevice
            try {
               // MY_UUID is the app's UUID string, also used by the server code
               tmp = device.createRfcommSocketToServiceRecord(MY_UUID);
            } catch (IOException e) { }
            mmSocket = tmp;

                    new Thread() {
                            public void run() {

                                    connect(device);
                                    Log.i("Conectado", "Conexion establecida");
                            };
                    }.start();
```

```
public void connect( BluetoothDevice device){
                // Cancel discovery because it will slow down the connection
    mBluetoothAdapter.cancelDiscovery();

    try {
       // Connect the device through the socket. This will block
       // until it succeeds or throws an exception
       mmSocket.connect();
    } catch (IOException connectException) {
       // Unable to connect; close the socket and get out
       try {
          mmSocket.close();
       } catch (IOException closeException) { }
       return;
    }

    // Do work to manage the connection (in a separate thread)
    //manageConnectedSocket(mmSocket);

        }
```

Y por ultimo esta el método que nos permite mandar los comandos por medio de bluetooth:

```
private void writeData(String data) {

                try {
            outStream = mmSocket.getOutputStream();
        } catch (IOException e) {
            Log.d("TAG", "Bug BEFORE Sending stuff", e);
        }

        String message = data;
        byte[] msgBuffer = message.getBytes();

            try {
                outStream.write(msgBuffer);

            } catch (IOException e) {
                  Log.d("TAG", "Bug while sending stuff", e);
            }
            Log.d("MAndado", data);
        }
```

4.3. Aplicación Android: Realización del Trazado

Para la realización del trazado, llamamos a la clase *DrawActivity,* que es la que nos permite dibujar. Para ello en el método onCreate establecemos como vista una instancia de la clase Touchevent:

```
@Override
protected void onCreate(Bundle savedInstanceState) {
        super.onCreate(savedInstanceState);
        touchevent = new TouchEvent(this, null);
        setContentView( touchevent );
}
```

Esta clase es la que nos permite dibujar en Android.

```
class TouchEvent extends View{

        final TouchEvent context = this;
        private Paint paint = new Paint();
        private Path path = new Path();



        public TouchEvent(Context context, AttributeSet attrs) {
                super(context, attrs);
                paint.setAntiAlias(true);
                paint.setStrokeWidth(8f);
                paint.setColor(Color.BLACK);
                paint.setStyle(Paint.Style.STROKE);
                paint.setStrokeJoin(Paint.Join.ROUND);
               }
```

Y en su constructor es donde establecemos, el tipo de dibujo que queremos, el color, el grosor, etc...

Pero el método mas interesante es el siguiente:

```
@Override
public boolean onTouchEvent(MotionEvent event) {
        float eventX = event.getX();
        float eventY = event.getY();

        switch (event.getAction()) {

            //Aqui pulsamos por primera vez la pantalla
            case MotionEvent.ACTION_DOWN:
                path.moveTo(eventX, eventY);
                vectorX[0] = (int)eventX;
                vectorY[0] = (int)eventY;
                int e = (int) eventY;
                Log.i("vectorY[0] = ", String.valueOf(e));
                Log.i("vectorX[0] = ", String.valueOf( (int) eventX ));
                return true;

              //Dibujamos conforme nos vamos moviendo
            case MotionEvent.ACTION_MOVE:
                path.lineTo(eventX, eventY);
                setVector(eventX, eventY);
                break;

              //Mensaje de terminacion del recorrido
            case MotionEvent.ACTION_UP:
                showDialog();
                Log.i("Tamaño del vector = ", String.valueOf(i));
                Log.i("vectorY[tamaño]", String.valueOf( vectorY[i-1] ));
                Log.i("vectorX[tamaño]", String.valueOf( vectorX[i-1] ));
                break;

            default:
                return false;
        }

              // Schedules a repaint.
 invalidate();
  return true;
 }
```

Porque es donde se establece que cuando el usuario pulse la pantalla y se mueva por ella, las coordenadas de (x,y) se vayan guardando en un vector que vamos formando.

```
public void setVector(float eventX, float eventY){
        vectorX[i] = (int)eventX;
        vectorY[i] = (int)eventY;
        i++;
}
```

Luego este vector es el que mandamos de nuevo a la clase principal cuando hemos terminado de dibujar. Ya que cuando el usuario levanta el dedo, llamamos al método *showDialog*(), que nos permite volver a la activity principal.

```
public void showDialog(){
                // 1. Instantiate an AlertDialog.Builder with its constructor
                  AlertDialog.Builder builder = new AlertDialog.Builder(this.getContext());

                  // 2. Chain together various setter methods to set the dialog characteristics
                  builder.setMessage("¿Desea fijar este recorrido?")
               .setTitle("Recorrido")
              .setNegativeButton("Ok", new DialogInterface.OnClickListener() {
                          public void onClick(DialogInterface dialog, int id) {
                             // User clicked OK button
                           devuelve();

                          }
                        })
               .setPositiveButton("Repaint", new DialogInterface.OnClickListener() {
                  public void onClick(DialogInterface dialog, int id) {
                             // User clicked Repaint button
                           paint.clearShadowLayer();
                           path.reset();
                           reset();

                          }
                        })
                        ;

                  // 3. Get the AlertDialog from create()
                  AlertDialog dialog = builder.create();
                  dialog.show();
                  }
```

Aquí devolvemos el vector:

```
public void devuelve(){
        Intent result = new Intent();
        result.putExtra("vectorX", vectorX);
        result.putExtra("vectorY", vectorY);
        result.putExtra("tamano",  i);
        setResult(RESULT_OK, result);
        finish();
}
```

4.4. Algoritmo

De nuevo en la MainActivity, tenemos con nosotros el vector con las coordenadas del trazo, y procedemos a su análisis:

```
public void movimiento() {

		//m es el valor de salto
		int m = (int)((tamaño *40)/100);
		//Log.i("tamaño", String.valueOf(tamaño));
		Log.i("m", String.valueOf(m));

		int longitud = ((int)((tamaño/m) + 1));
		Log.i("longitud", String.valueOf(longitud));

		float velocidad = 0.77f;// metros por segundo (m/s)

		for (int i = 1; i < longitud; i++){

			Log.i("valor de i", String.valueOf(i));

			int altura  = vectorY[ m * (i) ] - vectorY[ m * (i-1)];
			int giro    = vectorX[ m * (i) ] - vectorX[ m * (i-1)];

			final int tiempo = Math.abs((int)((altura / velocidad)));

			Log.i("espacio", String.valueOf(altura));
			Log.i("tiempo", String.valueOf(tiempo));

			Log.i("giro", String.valueOf(giro));
			Log.i("vector[1]", String.valueOf( vectorX[m * (i)] ));
			Log.i("vector[0]", String.valueOf( vectorX[m*(i-1)] ));


			//Movimiento hacia delante
			if ( altura < 0){
				if(giro < 0){
					try{
						dataToSend = "1";
						writeData(dataToSend);
						writeData("2");
						Thread.sleep(tiempo);
						dataToSend = "0";
						writeData(dataToSend);
```

```
                                        Log.i("Enviamos un 1", "Adelante");
                                    } catch(Exception e){}
                                } else {
                                    try{
                                    dataToSend = "1";
                                    writeData(dataToSend);
                                    writeData("3");
                                    Thread.sleep(tiempo);
                                    dataToSend = "0";
                                    writeData(dataToSend);

                                    Log.i("Enviamos un 1", "Adelante");
                                    } catch(Exception e){}
                                }

                            }
                            //Movimiento hacia atras
                            else{
                                if(giro < 0){
                                    try{
                                        dataToSend = "4";
                                  writeData(dataToSend);
                                  writeData("2");
                                  Thread.sleep(tiempo);
                                        dataToSend = "0";
                                        writeData(dataToSend);
                                        Log.i("Enviamos un 4", "Atras");
                                    } catch(Exception e){}
                                } else {
                                    try{
                                        dataToSend = "4";
                                  writeData(dataToSend);
                                  writeData("3");
                                  Thread.sleep(tiempo);
                                        dataToSend = "0";
                                        writeData(dataToSend);
                                        Log.i("Enviamos un 4", "Atras");
                                    } catch(Exception e){}
                                }
                            }

                        }

                    }
```

Vamos descomponiendo el movimiento en puntos, y vamos restándolos con el siguiente, para saber si el movimiento es adelante/atrás, y si tenemos que girar a la derecha o girar a la izquierda.

Además medimos la velocidad constante del vehículo, y sacamos que era aproximadamente de unos 0.77 m/s, con lo que sabiendo este dato, y calculando la distancia entre dos puntos, podemos sacar el tiempo que el coche necesita esta moviéndose.

$$s = v * t$$

Capítulo 5: Punto de vista del usuario

5.1. Introducción

En este capítulo veremos como funciona todo el sistema desde el punto de vista de un usuario que maneja el coche radiocontrol con la aplicación.

5.2. Iniciamos la APP

Nuestra aplicación tiene el nombre de BlueCar, debido a que manejamos el coche a través del bluetooth.

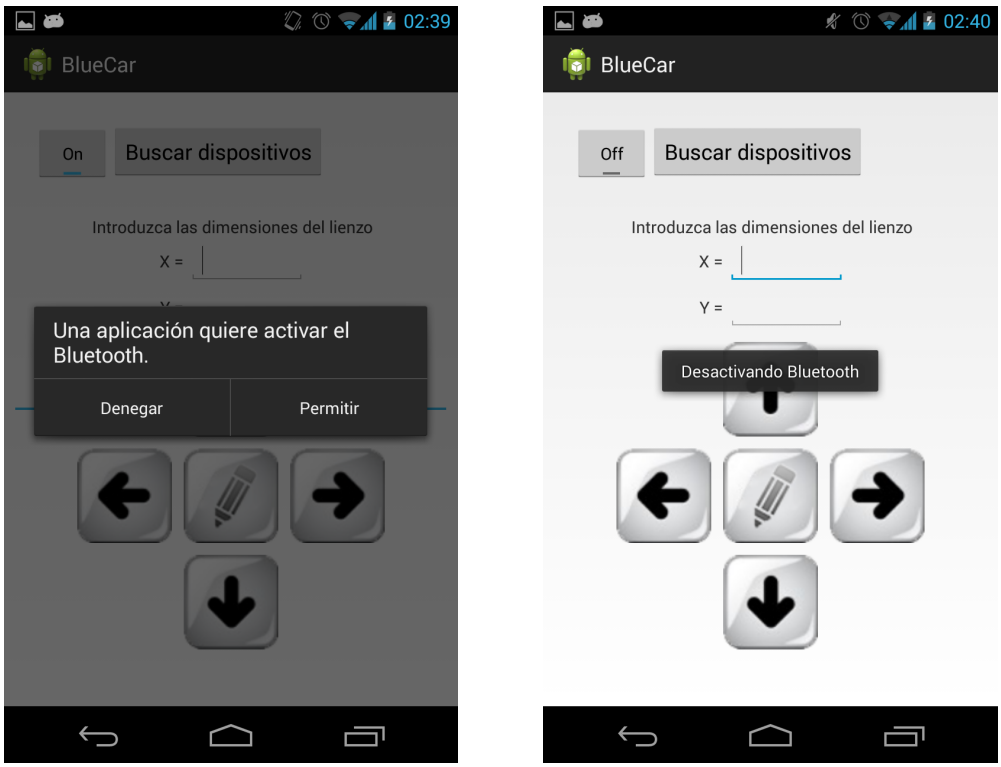
Al instalar en nuestro teléfono, versión de Android 4.3 (), veremos que nos aparece de la siguiente forma:

Figura 22: Aplicación BlueCar

Si la iniciamos nos encontramos con la siguiente pantalla:

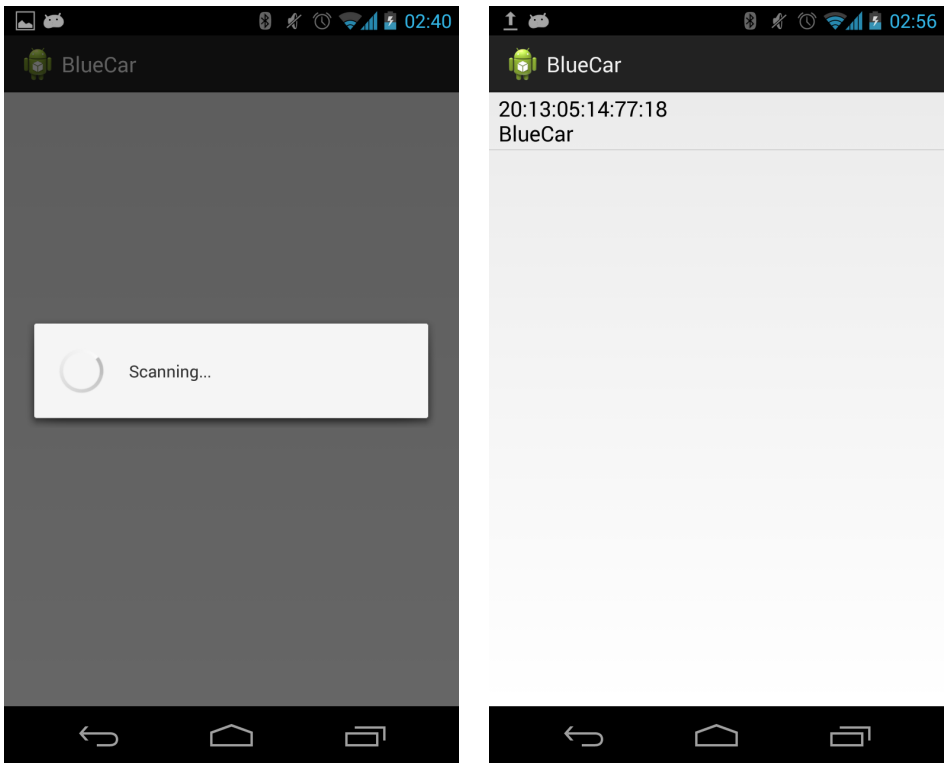
Figura 23: Pantalla principal

Si pulsamos el botón de On/Off, conseguiremos activar o desactivar el bluetooth de nuestro dispositivo móvil.



Figuras 24 y 25: Activando BT

Una vez activado tenemos que pulsar sobre “Buscar dispositivos”, este botón lo que nos hace es buscar otros dispositivos bluetooth a nuestro alrededor, para poder conectarnos con ellos:



Figuras 26 y 27: Escaneando dispositivos BT

Seleccionamos el dispositivo con el que queremos conectarnos, en nuestro caso con “BlueCar”, y automáticamente la aplicación no vuelve a la pantalla principal y se conecta con el dispositivo.

Para comprobar que la conexión se ha realizado correctamente podemos observar que la luz de nuestro módulo bluetooth ha dejado de parpadear, y ahora se mantiene fija.

Desde la pantalla principal, podemos controlar el coche con los cuatro botones de direcciones que disponemos, y podemos hacerlo avanzar o retroceder, y hacerlo girar a nuestro gusto.

5.3. Dibujar el trazo

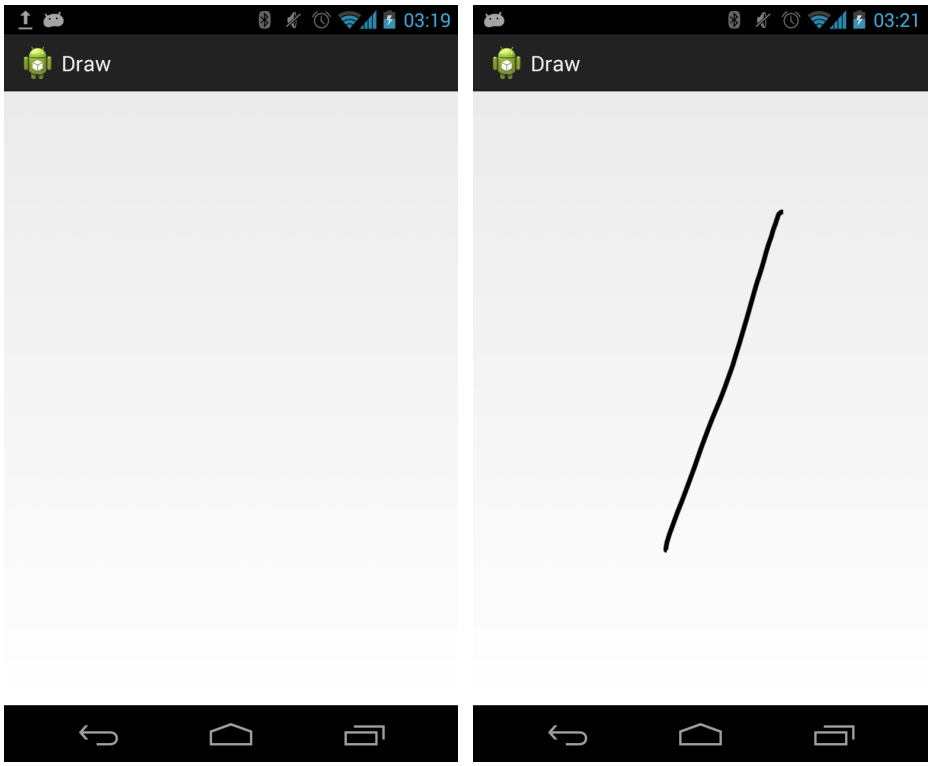
Pero también tenemos la posibilidad de dibujar el trazo que queremos que imite el vehículo, para ello tenemos que pulsar el botón con el icono del lápiz. Pero antes de hacerlo tenemos que fijar unas dimensiones para nuestro lienzo de dibujo, de forma que luego el recorrido del coche se ajuste a esas dimensiones.

Por eso antes de pulsar sobre el botón, introducimos los valores deseados para la X y para la Y, ambos en metros.

Figuras 28: Establecer los valores X,Y

Si pulsamos en cambio en el botón del centro, la aplicación nos lleva a otra ventana en la que se nos permite dibujar el recorrido que queremos que siga nuestro vehículo.

Conforme va nuestro dedo recorriendo la pantalla, se va trazando el recorrido que el vehículo tratará de imitar.



Figuras 29 y 30: Trazar recorrido

Una vez que levantamos el dedo de la pantalla nos sale un mensaje:

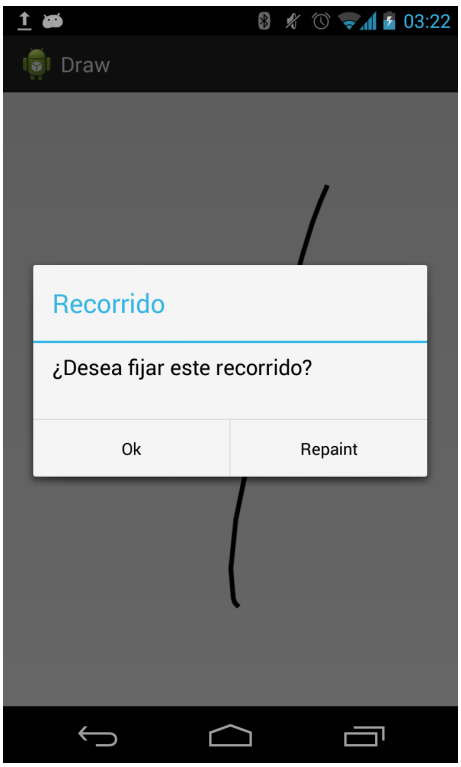


Figura 31: Confirmar recorrido

En el caso de que pulsemos “Repaint”, nos vuelve a la misma pantalla para que podamos volver a dibujar el trazado que deseamos.

En caso de que pulsemos “Ok”, la aplicación vuelve de nuevo a la Activity principal, donde se ejecuta el algoritmo que interpreta el trazo marcado, y el coche comienza a recorrer el dibujo

Capítulo 6: Conclusiones y líneas futuras

6.1. Introducción

En este último capitulo explicaremos las conclusiones sacadas del proyecto realizado, y de que modificaciones podrían incluirse en un futuro con el objetivo de mejorar el trabajo creado.

6.2. Conclusiones

En este proyecto hemos intentado que un coche radiocontrol convencional, sea capaz de realizar el mismo movimiento que puede trazar un pincel sobre un lienzo en blanco.

Lo primero que realizamos fue sustituir el mando teledirigido del vehículo por unos botones virtuales, de tal forma que comprobamos que teníamos control total sobre el movimiento.

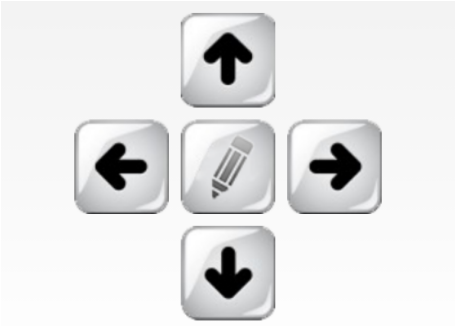


Figura 32: Botones virtuales

Después pasamos a dirigir el movimiento mediante un trazo dibujado en la pantalla. Aquí es donde mas dificultades encontramos debido a que era fácil realizar un momento hacia delante o hacia atrás, pero no tan sencillo era hacer que el coche girase. Esto es debido a que el movimiento de las ruedas es o giradas completamente hacia un extremo o rectas, no tienen posición intermedia, y por lo tanto no se pueden girar unos determinados grados.

Esto dificulta notablemente, que el vehículo pueda seguir una determinada curva dibujada, debido a que el radio de giro va a ser siempre el mismo. Por lo que en la realidad cuando queremos trazar un movimiento con muchos giros, o curvas muy cerrados, el resultado se aleja bastante del movimiento deseado.

Figura 33: Diferentes curvas

Se deberían añadir, algún tipo de control en el momento en que se pinta el trazo, para asegurarnos de que el vehículo es capaz de realizar ese movimiento correctamente. Y que no permita al usuario realizar trazados imposibles de luego llevar a cabo.

Figura 34: Trazo imposible

6.3. Líneas Futuras

Una posible línea de investigación futura podría ser añadir a nuestro vehículo radiocontrol un transistor que regule la velocidad, de tal forma que la velocidad no sea constante en todo momento.

Esto nos supondría una gran ventaja a la hora de realizar curvas muy pronunciadas, ya que se podría reducir la velocidad del dispositivo en el momento de realizar el giro. Pudiendo ampliar el rango de movimiento del vehículo.

Otra posible línea de investigación podría ser dotar al vehículo con sensores de proximidad, para que realice el trazo que le indicamos, pero si además se encuentra en su camino con algún obstáculo, sea capaz de detectarlo y corregir su movimiento para poder esquivarlo.

O la de añadirle video con una pequeña cámara, que introducimos en el vehículo, y así el usuario final aparte de comprobar como se mueve el vehículo y realiza el movimiento que el ha predeterminado, también podría recibir una señal de video como si estuviese dentro del coche, moviéndose con el.

Capítulo 7: Referencias

Aquí encontraremos algunos de los enlaces consultados durante la elaboración del documento.

Titulo: “Make Arduino Bots and Gadgets” , Editorial Oreilly, Edición Marzo 2011

http://androideity.com/2011/07/03/que-es-android/

http://es.wikipedia.org/wiki/Android

http://androideity.com/2011/07/04/arquitectura-de-android/

http://developer.android.com/develop/index.html

http://es.wikipedia.org/wiki/Bluetooth

http://www.masadelante.com/faqs/que-es-bluetooth

http://www.mobot.es/MobotBTCar.html

http://www.arduino.cc/es/

http://es.wikipedia.org/wiki/Arduino

http://www.arduteka.com/arduino/